Sommatore a 3 input (v2)

Stefano Scarcelli & Michele De Fusco

27 Dic 2023

Contents

1 Analisi progettuale

1.1 Analisi preliminare

L'obbiettivo è quello di costruire un circuito in grado di sommare 3 numeri a *n-bit* (*2's complements*) e restituirne il risultato. Sia gli input che gli output devono essere sincronizzati tramite l'uso di registri.
L'idea è quella di usare un sommatore

1.2 Struttura del progetto

La struttura del progetto sarà fatta da un modulo che implementa i **registri** e un'altro dove viene implementato il circuito finale (**Three adder**).

2 Implementazione

2.1 Register n-bit

L'implementazione dell'**Register n-bit** è la medesima del progetto precedente (v1).

2.1.1 Codice VHDL

```
----------------------------------------------------------------------------------
-- Company: UNICAL
-- Engineer: Stefano Scarcelli
--
-- Create Date: 04.12.2023 14:57:11
-- Design Name: ---
-- Module Name: Register_n - Version1
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Parametric n-bit register
--
-- Dependencies: None
--
-- Revision: 1
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Register_n is
    generic (n : integer := 8);
    Port (CLK, Clear : in STD_LOGIC;
          D : in STD_LOGIC_VECTOR (n-1 downto 0);
          Q : out STD_LOGIC_VECTOR (n-1 downto 0));
end Register_n;

architecture Version1 of Register_n is

begin
    process(CLK, Clear) begin
        if (Clear = '1') then
            Q <= (others => '0');
        elsif rising_edge(CLK) then
            Q <= D;
        end if;
    end process;

end Version1;
```

2.1.2 Sintesi

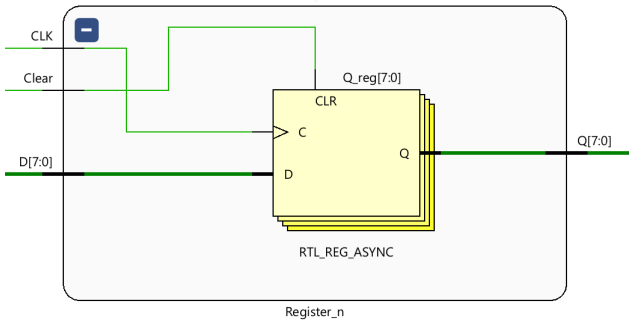

Figure 1: **Registro** a 8bit.

2.2 Three adder (main)

2.2.1 Codice VHDL

```
----------------------------------------------------------------------------------
-- Company: UNICAL
-- Engineer: Stefano Scarcelli, Michele De Fusco
--
-- Create Date: 04.12.2023 15:46:04
-- Design Name: ---
-- Module Name: Three_Adder - Version1
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Main file of the project
--
-- Dependencies: Register_n.vhd, Synched_adder.vhd
--
-- Revision: 1
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Three_Adder is
    generic (n : integer := 8);
    Port (A : in STD_LOGIC_VECTOR (n-1 downto 0);
          B : in STD_LOGIC_VECTOR (n-1 downto 0);
          C : in STD_LOGIC_VECTOR (n-1 downto 0);
          R : out STD_LOGIC_VECTOR (n+1 downto 0);
          CLK : in STD_LOGIC;
          Clear : in STD_LOGIC);
end Three_Adder;

architecture Version1 of Three_Adder is
    component Synched_adder
        generic (n : integer := 8);
        Port (A, B : in STD_LOGIC_VECTOR (n-1 downto 0);
              R : out STD_LOGIC_VECTOR (n downto 0);
              CLK, Clear : in STD_LOGIC);
    end component;
```

```
    component Register_n
        generic (n : integer := 8);
        Port (CLK, Clear : in STD_LOGIC;
              D : in STD_LOGIC_VECTOR (n-1 downto 0);
              Q : out STD_LOGIC_VECTOR (n-1 downto 0));
    end component;
    component Adder
        generic (n : integer := 8);
        Port (A, B : in STD_LOGIC_VECTOR (n-1 downto 0);
              R : out STD_LOGIC_VECTOR (n downto 0));
    end component;
    signal Rc: STD_LOGIC_VECTOR (n-1 downto 0);
    signal Rs,RCext: STD_LOGIC_VECTOR (n downto 0);
    signal RSF: STD_LOGIC_VECTOR (n+1 downto 0);
begin
    RCext <= Rc(n-1) & Rc;
    RegC: Register_n generic map(n) port map(CLK, Clear, C, Rc);

    SA1: Synched_adder generic map(n) port map(A, B, Rs, CLK, Clear);
    SA2: Synched_adder generic map(n+1) port map(Rs, RCext, RSF, CLK, Clear);

    RegSF:Register_n generic map(n+2) port map(CLK, Clear, RSF, R);

end Version1;
```

2.2.2 Sintesi

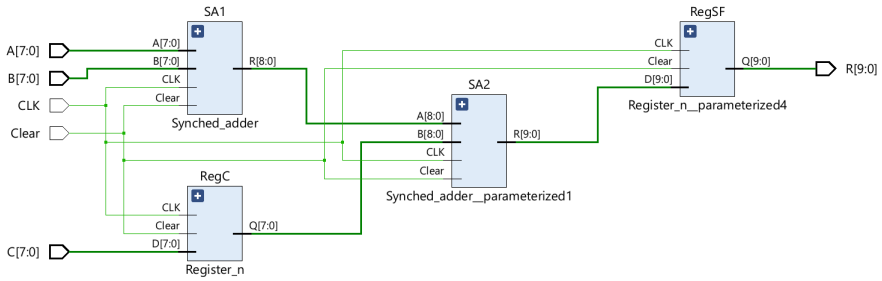

Figure 2: Circuito principale.

2.3 Time constraint

In più abbiamo aggiunto un *time constraint* sul segnale di clock con un periodo di **5ns**.

```
create_clock -period 5.000 -name main_clock
-waveform {0.000 2.500} [get_nets CLK]
```

2.3.1 Analisi del Time constraint

Con l'implementazione del codice abbiamo verificato (*Timing summary routed*) che il **Time constraint** fosse valido, valutando di aver inserito un periodo di clock di superiore rispetto al minimo tollerabile dal circuito di **1.958ns**, rispetto dai **1.958ns** del progetto precedente (v1).

2.4 Risorse

Name	Slice LUTs (53200)	Slice Registers (106400)	Slice (13300)	LUT as Logic (53200)	Bonded IOB (125)	BUFGCTRL (32)
N Three_Adder	17	51	12	17	36	1
RegC (Register_n)	0	8	3	0	0	0
RegSF (Register_n__parameterized4)	0	10	5	0	0	0
SA1 (Synched_adder)	8	16	8	8	0	0
SA2 (Synched_adder__parameterized1)	9	17	9	9	0	0

Figure 3: Risorse usate dai vari componenti.

2.4.1 Potenza

3 Testing

Per il test abbiamo deciso di usare il medesimo test del progetto precedente (v1).

3.1 Codice test test-bench

Resource	Utilization	Available	Utilization %
LUT	17	53200	0.03
FF	51	106400	0.05
IO	36	125	28.80

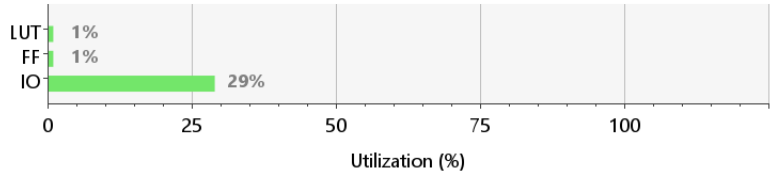

Figure 4: Riassunto delle risorse utilizzate.

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer: Stefano Scarcelli, Michele De Fusco
--
-- Create Date: 12.12.2023 11:20:24
-- Design Name: ---
-- Module Name: Sim_Add - Version1
-- Project Name: ThreeNumbersAdder
-- Target Devices: xc7z020clg400-2
-- Tool Versions: 2023.2
-- Description: Main simulation
--
-- Dependencies: Three_Adder.vhd
--
-- Revision: 2
-- Revision 1.0 - Implementation
-- Additional Comments: ---
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_arith.ALL;

entity Sim_Add is
    generic (n : integer := 8);
end Sim_Add;

architecture Version1 of Sim_Add is
```

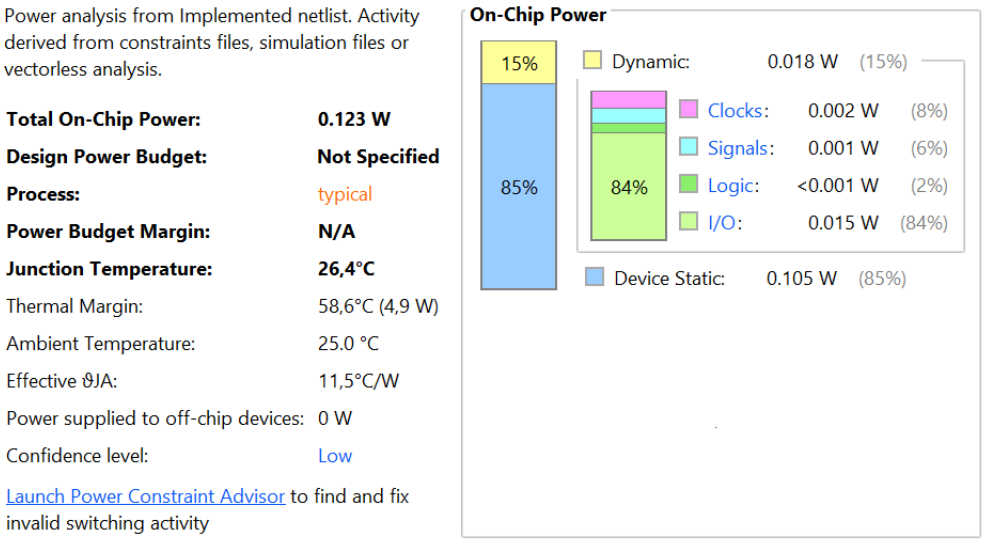

Figure 5: Riassunto del report della potenza del circuito.

```
    component Three_Adder
        --generic (n : integer := 8);
        Port (A : in STD_LOGIC_VECTOR (n-1 downto 0);
              B : in STD_LOGIC_VECTOR (n-1 downto 0);
              C : in STD_LOGIC_VECTOR (n-1 downto 0);
              R : out STD_LOGIC_VECTOR (n+1 downto 0);
              CLK : in STD_LOGIC;
              Clear : in STD_LOGIC);
    end component;
    Signal IA, IB, IC : STD_LOGIC_VECTOR (n-1 downto 0);
    Signal ORR : STD_LOGIC_VECTOR (n+1 downto 0);
    signal clk, clear : STD_LOGIC;
    constant T : time := 10 ns;
begin
    TA : Three_Adder port map(IA, IB, IC, ORR, clk, clear);
    process begin
        clk <= '0';
        wait for T/2;
        clk <= '1';
        wait for T/2;
    end process;

    process begin
        wait for 100 ns;
```

```
        clear <= '1';
        wait for T;
        clear <= '0';
        wait for T/4;

        -- (-1)+(-1)+(-1)=-3
        IA <= (others=>'1');
        IB <= (others=>'1');
        IC <= (others=>'1');
        wait for T;

        -- 127+127+127=381
        IA(n-1) <= ('0');
        IA(n-2 downto 0) <= (others=>'1');
        IB(n-1) <= ('0');
        IB(n-2 downto 0) <= (others=>'1');
        IC(n-1) <= ('0');
        IC(n-2 downto 0) <= (others=>'1');
        wait for T;

        -- (-64)+1+(-63)=0
        IA(n-1 downto n-2) <= (others=>'0');
        IA(n-3 downto 0) <= (others=>'1');
        IB(n-1 downto 1) <= (others=>'0');
        IB(0) <= ('1');
        IC(n-1 downto n-2) <= (others=>'1');
        IC(n-3 downto 0) <= (others=>'0');
        wait for T;

        -- Waiting results
        wait for T*2;
    end process;

end Version1;
```

3.2 Risultati

I risultati delle simulazioni confermano il funzionamento del circuito come previsto.

3.2.1 Simulazione comportamentale

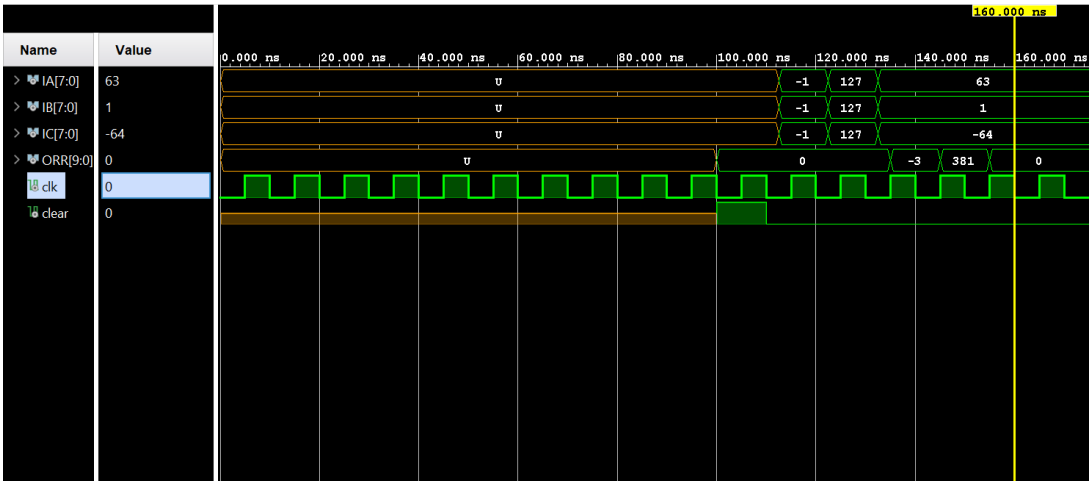

Figure 6: Grafico temporale simulazione comportamentale.

3.2.2 Simulazione post sintesi

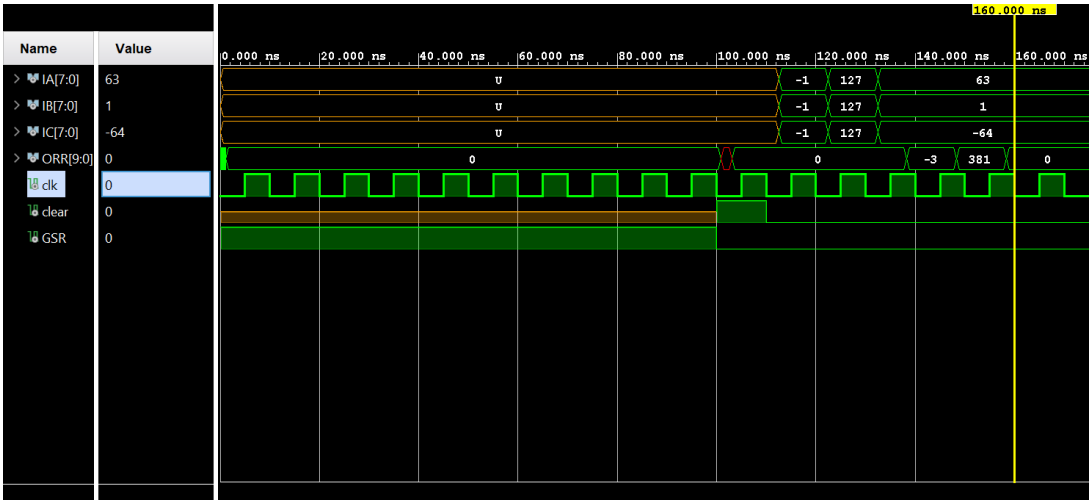

Figure 7: Grafico temporale simulazione post sintesi.

3.2.3 Simulazione post implementazione

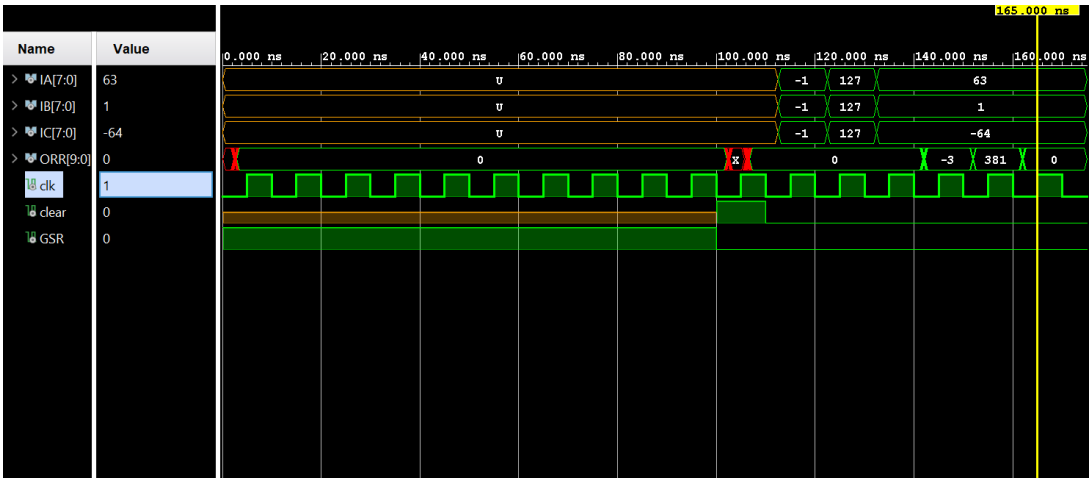

Figure 8: Grafico temporale simulazione post implementazione.